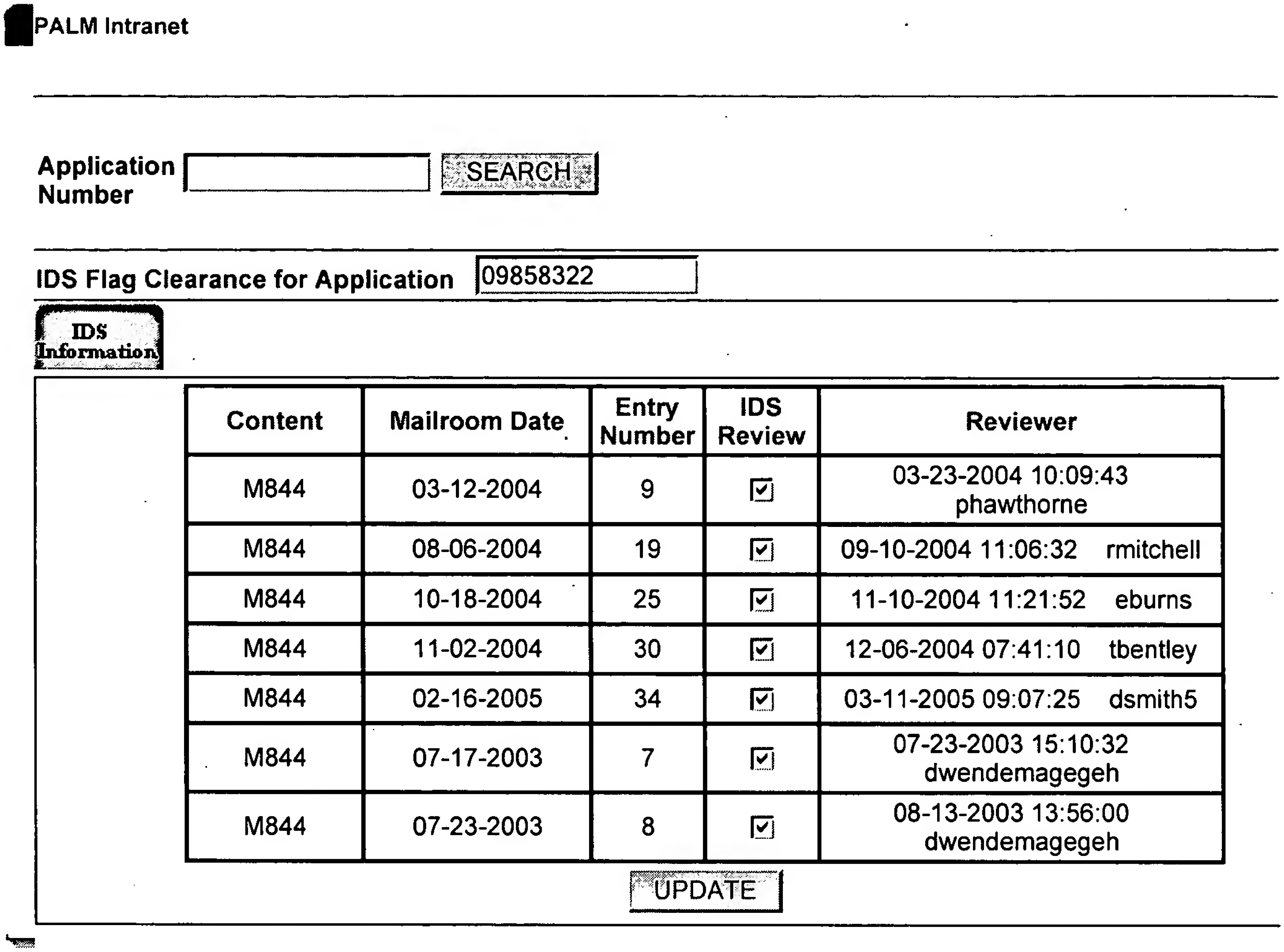


PALM Intranet

Application Number [ ] SEARCH

IDS Flag Clearance for Application 09858322

Content	Mailroom Date	Entry Number	IDS Review	Reviewer
M844	03-12-2004	9	☑	03-23-2004 10:09:43 phawthorne
M844	08-06-2004	19	☑	09-10-2004 11:06:32 rmitchell
M844	10-18-2004	25	☑	11-10-2004 11:21:52 eburns
M844	11-02-2004	30	☑	12-06-2004 07:41:10 tbentley
M844	02-16-2005	34	☑	03-11-2005 09:07:25 dsmith5
M844	07-17-2003	7	☑	07-23-2003 15:10:32 dwendemagegeh
M844	07-23-2003	8	☑	08-13-2003 13:56:00 dwendemagegeh

Refine Search

Search Results -

Terms	Documents
L14 and (sell$ or sale or commerce or transact$)	2

Database:

US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L15

Refine Search

Recall Text   Clear   Interrupt

Search History

DATE: Monday, October 17, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; THES=ASSIGNEE; PLUR=YES; OP=OR		
L15	L14 and (sell$ or sale or commerce or transact$)	2	L15
L14	L13 and (database$ with pointer$)	2	L14
L13	l12 or l11 or l10	107	L13
	DB=USPT; THES=ASSIGNEE; PLUR=YES; OP=OR		
L12	('6366910'| '6578014'| '6078891')[URPN]	36	L12
L11	(5987460 | 5513102 | 4789928 | 5490252 | 5557780 | 5848409 | 5802493 | 5315093 | 6012045 | 4903201 | 5590196 | 5590197 | 4799156 | 5608874 | 5103392 | 5604804 | 5592378 | 5722418 | 5794207 | 6011862 | 5062074 | 4943963 | 4412287 | 6272636 | 5787422 | 5835896 | 5499385 | 5253345 | 4951196 | 6035288 | 5895454 | 5592560 | 6029195 | 5845265 | 3956615 | 5251131 | 5758328 | 5897638 | 5168446 | 4438824 | 5331544 | 5950172 | 5924082 | 00H1708 | 5202977 | 5521363 | 5491473 | 4319336 | 6195652 | 5557518 | 4972504 | 6038560 | 4355372 | 5490060 | 5835905 | 5687322 |	68	L11

	6216112 | 5905975 | 6260024 | 4340810 | 5488725 | 5832460 | 5442778 | 5077665 | 6023685 | 5920854 | 5684965 | 5907617 | 5890138)![PN]		
L10	('6366910'| '6578014'| '6078891')[PN]	3	L10
L9	L6 and (database$ with pointer$)	0	L9
L8	L7 and (attribut$ and pointer and (item$ or product$))	1	L8
L7	l1 or l5 or L6	3	L7
L6	6578014.pn.	1	L6
L5	6078891.pn.	1	L5
L4	L1 and (attribut$ and pointer and (item$ or product$))	0	L4
L3	L1 and (attribut$ same (item$ or product$))	0	L3
L2	L1 and (attribut$ with product$)	0	L2
L1	6366910.pn.	1	L1

END OF SEARCH HISTORY

L14: Entry 2 of 2 File: USPT Sep 2, 2003

US-PAT-NO: 6615258
DOCUMENT-IDENTIFIER: US 6615258 B1

TITLE: Integrated customer interface for web based data management

DATE-ISSUED: September 2, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Barry; B. Reilly	Colorado Springs	CO		
Chodoronek; Mark A.	Centreville	VA		
DeRose; Eric	Falls Church	VA		
Gonzales; Mark N.	Manassas	VA		
James; Angela R.	Chevy Chase	MD		
Levy; Lynne	Herndon	VA		
Tusa; Michael	Ridgefield	CT		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Worldcom, Inc.	Clinton	MS			02

APPL-NO: 09/159505 [PALM]
DATE FILED: September 24, 1998

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATIONS The following patent application is based on and claims the benefit of U.S. Provisional Patent Application Ser. No. 60/060,655 filed Nov. 1, 1997, entitled INTEGRATED CUSTOMER INTERFACE SYSTEM FOR COMMUNICATIONS MANAGEMENT.

INT-CL: [07] G06 F 15/173

US-CL-ISSUED: 709/223; 709/229
US-CL-CURRENT: 709/223; 709/229

FIELD-OF-SEARCH: 709/223-224, 709/217-219, 709/229, 713/151, 713/154, 705/40-44, 705/63, 705/75, 705/77, 379/112, 379/114, 379/140, 379/201, 379/265

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#
End of Result Set

□ Generate Collection Print

L14: Entry 2 of 2 File: USPT Sep 2, 2003

DOCUMENT-IDENTIFIER: US 6615258 B1
TITLE: Integrated customer interface for web based data management

Detailed Description Text (217):
The online invoicing includes a DB2 database subsystem residing in a NOR4 mainframe. The subsystem further includes an object database and an index database. An online invoicing store process 1460 loads the compressed document to an online invoicing object database and an online invoicing index load process 1480 stores index pointers to each document in the index database. An audit check is executed to ensure that the correct number of documents are added to the online invoicing databases during the object load and index load processes.

US Reference Patent Number (184):
6078891

CLAIMS:

25. The integrated data management system as claimed in claim 23, wherein the database of image files further includes an index database, and the invoice server further includes an index load device for storing index pointers pointing to the compressed documents.

57. The method as claimed in claim 55, wherein the method further comprises: storing index pointers in an index database, the index pointers pointing to the documents in the object database.

Previous Doc Next Doc Go to Doc#

Feb. 28, 2002.

"Cryptography and the Internet", www.echonyc.com/.about.ysue/crypt.html, 1995.

Lee et al., "Supporting Multi-User, Multi-Applet Workspaces in CBE", Computer Supported Cooperative Work 1996, Cambridge, MA.

"Netscape 2.0 Beta Hip or Hype?", www.plant.net.au/innovations/20beta.html, Planet Internet, 1995.

Meterology; Databases, "Inforonics offers controlled access to Web Meterology", Information Today, Apr. 1997, vol. 14 Issue 4, p53, 2p. This article reports that Inforonics has developed a controlled access gateway to MGA (Meteorological and Geoastrophysica).

Rosen,Michele, BPCS steps into new millennium, Midrange Systems; Spring House; May 10, 1996. This article informs about the new release of BPCS Client/Server Software as the most extensive upgrade of the product since 1980s. It incorporates onject tech.

Inoue et al., "Secure Mobile IP Using Security Primitives", IEEE 1997.

Biggs, M., "Help for the Web enhances customer support, reduces help desk load" Inforworld, Jun. 16, 1997, v. 19, No. 24, pp. 82+.

Burch, B., "AT&T, MCI to release new management tools", Network World, Jan. 17, 1994, p. 19.

Low, C., "Integrating Communication Services", IEEE Communication Magazine, Jun. 1997, pp. 164-169.

"McAfee's New `Self-Service` Help Desk Web Suite Makes PCs Help Desk-Ready", Newswire Association Inc., Oct. 13, 1997.

Niemeyer, R., "Using Web Technologies in Two MLS Environments: A Security Analysis." IEEE, pp. 205-214, 1997.

Porter, T., "MCI offers tracking system: Direct Dispatch lets users eye problems remotely", Service News, Apr. 1994, p. 17.

Shklar, L., et al., "MetaMagic: Generating Virtual Web Sites Through Data Modeling," http://www.scope.gmd.de/info/www6/posters/714/poster714.html.

Vizard, M. et al., "MCI to Pilot Convergence Billing Service", InfoWorld, v. 18, Issue 37, Sep. 9, 1996.

Yager, T., "Mixed Messages", UNIX Review, v. 16, n. 2, p. 29, Feb. 1998.

"Carriers Improve Net Management Services", Communications Week, May 2, 1994, p. 74.

"Network management; new software platform enhances network management capabilities; MCI ServiceView offers greater cost savings, increased flexibility.", Product Announcement, Edge, Oct. 2, 1995, on & about AT&T, v. 10, n. 375, p. 11(1).

He, Taniguchi, "Internet Traffic Control and Management Architecture", IEEE, Oct. 22-24, 1998, pp. s46-03-1-s46-03-5.

Sixth International Conference on Network Protocols, IEEE, Technical Communication Services, Oct. 13-16, 1998, Table of Contents.

Markovich, Robert, "WAN Service Level Management Could Keep Your Feet Out of the Fire, Ensure Carriers Dilligence", Network World, Jul. 7, 1997.

Computer Networks, Andrew S. Tanenbaum, pp. 410-412.

"XIIR6.3 (Broadway) Overview", http://www.x.org/broadway.htm.

"Stac Unveils Windows NT 4.0 and Web Browser Support in New ReachOut 7" http://www.stac.com/news/pressrel/pr_ro7_unveil.html.

"Help-Desk Market Seeks Suite Success", Computer Reseller News, Jan. 5, 1998, p. 49.

ART-UNIT: 2141

PRIMARY-EXAMINER: Jaroenchonwanit; Bunjob

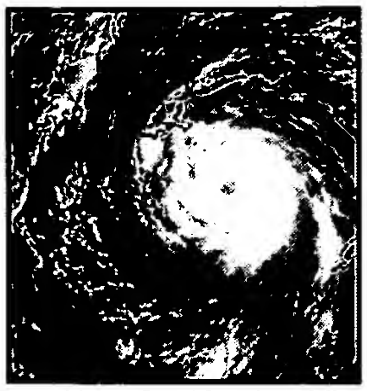
ABSTRACT:

An integrated data management system for providing data management services from an enterprise over the Internet is described. A user interface executable in a customer workstation authenticate the customer's access to the system and presents one or more data management services according a customer entitlement, for the customer to select. Client applications representing the data management services re initiated by the user interface in response to customer selection. Consequently, the customer is enabled at the customer site to request and receive the data management services according to the customer's entitlements in a secure Internet-based computing environment.

63 Claims, 44 Drawing figures

Previous Doc Next Doc Go to Doc#

9/858322



Anonymous User

Join

Login

Features

Lab Wear

Advanced Search

Related

HOW-TO Developers Developer's Guide

ZOPE

Get Zope

Achieve Zen

Newbies

CommunityRant

The Book

Building Websites

Hosting

EU Hosting

Information

Help!

PyReference

PyCookbook

FAQTs

Daily-URL!

Contact

hoı

Access Product Class Attributes By Metatype

Submitted by: **cduncan**
Last Edited: 2004-10-14

Av

Category: Python(External Method)

Description:

I developed a site where there are several content classes, each with a slightly differe

Being lazy and since many of the property fields were the same for all the classes, I c page template) that had all possible fields defined in it (about 8 or so). When it was d that were appropriate. The former is easy since I can just use the PropertyManager A since no object yet exists in Zope to look at (This was for creating dynamic TTW obj

The solution I came up with was a short external method that given a meta-type can i from it. This could be easily modified to get at any attributes of a class though. Note: not be necessary in their case anyhow.

The second caveat is that it can only access classes for objects that can be added in tł this, however, by getting all_meta_types from the application root instead of self.

Source (Text):

```
def getPropertyIds(self, meta_type):
    """Return a list of names of the property schema of
       a given meta-type
    """
    types = self.all_meta_types
    if callable(types): types = types()

    for t in types:
        if t['name'] == meta_type:
            klass = t.get('instance', None)
            # Modify the lines below to return other att:
            properties = getattr(klass, '_properties', []
            return map(lambda i: i['id'], properties)

    raise ValueError, 'Meta-type "%s" not found.' % meta_
```

Explanation:

As with any external method code, be sure you don't accidently cause security holes. You should avoid returning mutable attributes (such as lists and dictionaries) without first making a copy of them. The above code is safe (AFAICT ;^), since it just returns a list of strings created dynamically and not the whole original data structure from the class.

Comments:

It works fine for me by groovehunter - 2004-10-14

```
I call it from a ZWikiPage with:
<dtml-var "getPropertyIds('ZWiki Page')">
```

```
Example result:
['title', 'page_type', 'creator', ... blah ... 'subscrib
```

The Single UNIX ® Specification, Version 2

NAME

dbm_clearerr, dbm_close, dbm_delete, dbm_error, dbm_fetch, dbm_firstkey, dbm_nextkey, dbm_open, dbm_store - database functions

SYNOPSIS

```
#include <ndbm.h>

int dbm_clearerr(DBM *db);
void dbm_close(DBM *db);
int dbm_delete(DBM *db, datum key);
int dbm_error(DBM *db);
datum dbm_fetch(DBM *db, datum key);
datum dbm_firstkey(DBM *db);
datum dbm_nextkey(DBM *db);
DBM *dbm_open(const char *file, int open_flags, mode_t file_mode);
int dbm_store(DBM *db, datum key, datum content, int store_mode);
```

DESCRIPTION

These functions create, access and modify a database.

A **datum** consists of at least two members, **dptr** and **dsize**. The **dptr** member points to an object that is **dsize** bytes in length. Arbitrary binary data, as well as character strings, may be stored in the object pointed to by **dptr**.

The database is stored in two files. One file is a directory containing a bit map of keys and has **.dir** as its suffix. The second file contains all data and has **.pag** as its suffix.

The *dbm_open()* function opens a database. The *file* argument to the function is the pathname of the database. The function opens two files named and The *open_flags* argument has the same meaning as the *flags* argument of *open()* except that a database opened for write-only access opens the files for read and write access and the behaviour of the O_APPEND flag is unspecified. The *file_mode* argument has the same meaning as the third argument of *open()*.

The *dbm_close()* function closes a database. The argument *db* must be a pointer to a **dbm** structure that has been returned from a call to *dbm_open()*.

The *dbm_fetch()* function reads a record from a database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*. The argument *key* is a **datum** that has been initialised by the application program to the value of the key that matches the key of the record the program is fetching.

The *dbm_store()* function writes a record to a database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*. The argument *key* is a **datum** that has been initialised by the application program to the value of the key that identifies (for subsequent reading, writing or deleting) the record the program is writing. The argument *content* is a **datum** that has been initialised by the application program to the value of the record the program is writing. The argument *store_mode* controls whether *dbm_store()* replaces any pre-existing record that has the same key that is specified by the *key* argument. The application program must set *store_mode* to either DBM_INSERT or DBM_REPLACE. If the database contains a record that matches the *key* argument and *store_mode* is DBM_REPLACE, the existing record is replaced with the new record. If the database contains a record that matches the *key* argument and *store_mode* is DBM_INSERT, the existing record is not replaced with the new record. If the database does not contain a record that matches the *key* argument and *store_mode* is either DBM_INSERT or DBM_REPLACE, the new record is inserted in the database.

The sum of the sizes of a key/content pair must not exceed the internal block size. Moreover, all key/content pairs that hash together must fit on a single block. The *dbm_store ()* function returns an error in the event that a disk block fills with inseparable data.

The *dbm_delete()* function deletes a record and its key from the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*. The argument *key* is a **datum** that has been initialised by the application program to the value of the key that identifies the record the program is deleting.

The *dbm_firstkey()* function returns the first key in the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*.

The *dbm_nextkey()* function returns the next key in the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*. The *dbm_firstkey()* function must be called before calling *dbm_nextkey()*. Subsequent calls to *dbm_nextkey()* return the next key until all of the keys in the database have been returned.

The *dbm_error()* function returns the error condition of the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*.

The *dbm_clearerr()* function clears the error condition of the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm_open()*.

These database functions support key/content pairs of at least 1023 bytes.

The **dptr** pointers returned by these functions may point into static storage that may be changed by subsequent calls.

These interfaces need not be reentrant.

RETURN VALUE

The *dbm_store()* and *dbm_delete()* functions return 0 when they succeed and a negative value when they fail.

The *dbm_store()* function returns 1 if it is called with a *flags* value of DBM_INSERT and the function finds an existing record with the same key.

The *dbm_error()* function returns 0 if the error condition is not set and returns a non-zero value if the error condition is set.

The return value of *dbm_clearerr()* is unspecified .

The *dbm_firstkey()* and *dbm_nextkey()* functions return a key **datum**. When the end of the database is reached, the **dptr** member of the key is a null pointer. If an error is detected, the **dptr** member of the key is a null pointer and the error condition of the database is set.

The *dbm_fetch()* function returns a content **datum**. If no record in the database matches the key or if an error condition has been detected in the database, the **dptr** member of the content is a null pointer.

The *dbm_open()* function returns a pointer to a database structure. If an error is detected during the operation, *dbm_open()* returns a (**DBM** *)0.

ERRORS

No errors are defined.

EXAMPLES

None.

APPLICATION USAGE

The following code can be used to traverse the database:

```
for(key = dbm_firstkey(db); key.dptr != NULL; key = dbm_nextkey(db))
```

The *dbm_* functions provided in this library should not be confused in any way with those of a general-purpose database management system. These functions do not provide for multiple search keys per entry, they do not protect against multi-user access (in other words they do not lock records or files), and they do not provide the many other useful database functions that are found in more robust database management systems. Creating and updating databases by use of these functions is relatively slow because of data copies that occur upon hash collisions. These functions are useful for applications requiring fast lookup of relatively static information that is to be indexed by a single key.

The *dbm_delete()* function need not physically reclaim file space, although it does make it available for reuse by the database.

After calling *dbm_store()* or *dbm_delete()* during a pass through the keys by *dbm_firstkey()* and *dbm_nextkey()*, the application should reset the database by calling *dbm_firstkey()* before again calling *dbm_nextkey()*. The contents of these files are unspecified and may not be portable.

FUTURE DIRECTIONS

None.

SEE ALSO

open(), *<ndbm.h>*.

[Main Index | XSH | XCU | XBD | XCURSES | XNS]